Interoperable RNA-seq analysis in the cloud

Alexander Lachmann[1,2,3,✉], **Daniel J. B. Clarke**[1,2,3], **Denis Torre**[1,2,3], **Zhuorui Xie**[1,2], **and Avi Ma'ayan**[1,2,3]

[1] Department of Pharmacological Sciences, Icahn School of Medicine at Mount Sinai, One Gustave L. Levy Place, Box 1603, New York, NY 10029, USA
[2] Library of Integrated Network-based Cellular Signatures, Data Coordination and Integration Center (BD2K-LINCS DCIC)
[3] Knowledge Management Center for Illuminating the Druggable Genome (KMC-IDG)

RNA-Sequencing (RNA-Seq) is currently the leading technology for genome-wide transcript quantification. Mapping the raw reads to transcript and gene level counts can be achieved by different aligners. Here we report an in-depth comparison of transcript quantification methods. Our goal is the specific use of cost-efficient RNA-Seq analysis for deployment in a cloud infrastructure composed of interacting microservices. The individual modules cover file transfer into the cloud and APIs to handle the cloud alignment jobs. We next demonstrate how newly generated RNA-Seq data can be placed in the context of thousands of previously published datasets in near real time. With in-depth benchmarks, we identify suitable gene count quantification methods to facilitate cost-effective, accurate, and cloud-based RNA-Seq analysis service.

Correspondence: ***alexander.lachmann@mssm.edu***

Introduction

Genome-wide gene expression data from thousands of studies have been accumulating and made available for exploration and reuse through public repositories such as the Gene Expression Omnibus (GEO) (1). RNA-Seq is supplanting cDNA microarrays as the dominant technology due to competitive cost, increased sensitivity, ability to quantify splice variants, and improved reproducibility. The processing of RNA-Seq data remains a challenge due to demand in computational resources as well as the required technical skills needed to execute computational pipelines. The process of aligning raw reads to a reference genome can be performed with a multitude of alignment algorithms (2–6) as well as different genome assemblies and transcript annotations. In order to improve reproducibility, workflow languages have been introduced (7, 8). The Common Workflow Language (CWL) is a standard to describe data analysis workflows. One of the implementations of this standard is the Workflow Description Language (WDL). Pipeline frameworks such as Toil (9) build on top of workflow languages. They provide higher level programming interfaces to deploy and control workflows written in CWL. Toil provides an application program interface (API) for static and dynamically generated workflow specifications and integration with other microservices. Toil allows the recursive generation of new workflows during application runtime and can be deployed in the cloud while being agnostic to environments such as Amazon AWS and Azure Google Cloud. The genome analysis toolkit (GATK) (10) is a framework for the analysis of genomic data. GATK implements a collection of command-line tools of best practice approaches for a variety of tasks such as variant calling from raw sequencing reads. It provides extensive documentation and is based on workflows described in WDL. Galaxy (11) and Taverna (12) are examples of open source scientific workflow platforms. Galaxy and Taverna were created due to the growing need of biologists that lack programming and scripting experience to develop step-wise data manipulation instructions. Both platforms provide user interfaces. While Taverna is mainly meant to run on local instances, Galaxy can be installed locally or run on cloud environments. Both platforms support plugins which extend their functionality over time. As such, they rely on active community support. There are also commercial platforms that offer services similar to Galaxy (13–16).

Several attempts to reprocess extensive collections of publicly available gene expression data address the need for uniform processing for better interoperability (9, 17–19). In this context, we developed ARCHS4 (20). ARCHS4 processes most of the RNA-Seq data from GEO collected from human or mouse, supplying the largest repository of homogeneously processed RNA-Seq data with 206,643 mouse and 167,184 human samples as of June 2019. ARCHS4 has a scalable cloud computing infrastructure capable of adjusting to fluctuations in demand. Efficient implementation of storage and cloud compute instances are critical attributes for a cost-efficient and scalable solution. Here we present a new API driven implementation that orchestrates four components to allow users to align their RNA-Seq data in the cloud with an enhanced version of the ARCHS4 pipeline. Additionally, we review some of the most widely used alignment algorithms and benchmark their performance. Alignment-free or pseudo-alignment algorithms promise a significant cost advantage over standard alignment algorithms. However, there have been some concerns that the accuracy of alignment-free methods is not up to par with standard transcript quantification methods (21). Here we discuss the basic choices in the development of RNA-Seq processing pipelines to minimize cost while guaranteeing high-quality transcript abundance estimation. Additionally, we introduce some of the concepts that are key in the generation of extendable distributed workflow pipelines that are built on a set of micro-services. Such implementation allows the processing of thousands of FASTQ files from many users without the need for setting up user accounts and without usage limitations. Specifically, these simple techniques result in a cost efficiency that allows hosting of a near-free RNA-Seq processing service with an effective compute-cost of less than $100 for aligning more than 10,000 RNA-seq samples.

Pre-processing of RNA-Seq reads

Adapters are a common source of contamination of raw reads resulting in reduced alignment quality. Adapters are required for most sequencing protocols. After DNA fragmentation through sonication, specific oligonucleotide sequences are attached through ligation to the 5' and 3' ends of the fragments (22). These adapters are needed to barcode paired-end reads and for letting them adhere to the flowcell surface (23). The high number of different adapter trimming tools suggests that computational removal of adapter sequences is an important step in the pre-processing of reads before any alignment procedures can be performed (24–30). Since the adapter sequences will be read if the DNA fragment size is lower than the number of sequencing cycles, a certain fraction of reads can be expected to contain some unwanted adapter oligonucleotides. Due to the stochastic nature of the fragmentation process, the distribution of fragment sizes varies and will ultimately include shorter reads. It should be noted that adapter sequences are not found in the genome, and thus it is not possible to align them to the genome. It would be impossible to match contaminated reads to a genomic location if aligners would require perfect matches of the reads to the reference genome. Also, low-quality reads can be removed in this step or trimmed down to subsequences of high quality. A comprehensive comparison of a variety of nine trimming tools (26, 31–37) was presented in (24). The authors of this benchmarking application showed that trimming will result in a reduction of final read counts in RNA-seq studies while increasing the alignment success rate of surviving reads. The comparative analysis of the trimming tools does not directly show that the resulting gene counts are more accurate in representing transcript abundance. While adapter trimming has been shown to improve de novo assembly of genomes (38) modestly, there is evidence that rigorous trimming can have a negative effect (39). In the absence of studies confirming additional benefits, most pipelines applied to process large RNA-seq repositories omit this step all together (18, 20, 40).

Alignment algorithms

The primary tool needed to quantify the abundance of mRNA raw reads are aligners that can map nucleotide reads to a matching location on a reference sequence. Depending on the application these reference sequences are whole genomes or transcript sequences. If reads are aligned against a reference genome, the genome annotation is used to identify regions that overlap with known transcript sequences to quantify the abundance of mRNA. These algorithms also play a central role in applications such as peak calling of transcription factor binding sites or genomic variation analysis (41). A new class of algorithms, the pseudo-aligners, or alignment-free methods, were developed recently to improve the speed and memory footprint during the alignment process. Speed and efficient memory utilization are important since the alignment procedure can be expensive to perform on a large scale. Cost is also a major factor when processing RNA-Seq data. Alignment-free algorithms map reads directly to transcript sequences, reducing the size of the reference significantly. For transcript quantification, alignment-free methods have advantages and shortcomings. True aligners use indexes that enable the identification of sequence matches to a reference. A reference genome index is a data structure that is designed to speed up this process. The naive way of finding a matching sequence to a reference is to match every possible location of the reference. This approach is too slow when the reference has millions of nucleotides, and the location of multiple input sequences need to be identified. Most classic aligners such as Spliced Transcripts Alignment to a Reference (STAR) and Burrows-Wheeler alignment (BWA) transform the reference sequence into a suffix array, or in the case of BWA apply a burrows-wheeler transform. This index data structure saves the position of each prefix of a reference sequence (42–44) and this supports a fast search for arbitrary sequences. Most algorithms that support prefix indexing try to identify a maximally matching seed, or multiple seeds, from the input sequence to the reference in a seed mapping phase. Once a maximal match has been established, the seed can be extended locally in both directions allowing gaps and mismatches. This strategy is important as RNA-Seq reads can contain sequences that are produced through splicing of distant DNA sequences. Additionally, mismatches due to erroneous sequencing can occur as a technical artifact. Especially the ends of reads are prone to higher sequencing errors resulting in mismatches. Seed matching is less sensitive to these errors as it can find anchor seeds in any position of the input sequence. Most true aligners support a 2-pass alignment mode. In this mode, the algorithms use alignments from the first alignment pass to identify potential fusion locations and splice junctions to improve alignment accuracy. While the basic strategy of these algorithms is essentially the same, the details of the alignment process can result in significant differences. STAR and Hierarchical Indexing for Spliced Alignment of Transcripts (HISAT2) are two fairly recent implementations with significant advantages in performance. Due to a layered index structure for local and global indexing, HISAT2 can maintain a small memory footprint compared to STAR. The alignment-free aligners, pseudo-aligners, or lightweight-aligners, such as kallisto and Salmon, take a different strategy that is not based on matching arbitrary substrings of an input sequence to a reference. Instead, pseudo-alignment algorithms use hashing to compress the input sequence into a unique pointer to a transcript. When applying a hash function to a sequence, the input sequence can be mapped to an output hash value. Using a hash-map, a hashed sequence can be used as a key that is directly pointing to a stored value. The lookup can be performed in $O(1)$. For the use of hash-maps in context of alignment, all subsequences of the reference are stored with the corresponding position of origin in the reference. This method does, however, run into technical limitations. The read length varies between experiments, but a hash-map will only work for a predefined sequence length. Additionally, hash-maps do not support mismatches; hence, even small changes to the input sequence will, in most cases, result in

completely different hash values (45). Pseudo-aligners rely on k-mers to leverage the speed advantage of direct sequence mapping. The idea is to apply k-mer shredding, transforming each read into a set of overlapping sub-sequences of fixed length. For example, shredding the sequence $ACTGC$ to k-mers of length three results in the set $\{ACT, CTG, TGC\}$. The k-mers represent a sliding window of a fixed length over the sequence. To identify the abundance of transcripts, hash-tables are only needed to index known transcript sequences. A naive algorithm could count the number of k-mers that map to a transcript. Instead of counting k-mers, a refinement relies on the notion of abundance estimation using expectation maximization. The likelihood function for RNA-Seq is defined by:

$$L(\alpha) = \prod_{f \in F} \sum_{t \in T} y_{f,t} \frac{\alpha_t}{l_t}$$

where F is a set of k-mers, T is the set of transcripts, l_t is the length of transcript t and $y_{f,t}$ denotes the compatibility matrix. $y_{f,t}$ is 1 when a k-mer is a subsequence of transcript t. If the k-mer f is not unique, multiple entries of f can be 1. A k-mer is not compatible when it is not a subsequence of t and in this case $y_{f,t} = 0$. α_t are the transcript probabilities, which are proportional to the transcript abundances. This likelihood function is described in more detail by the publication describing kallisto and Salmon (46, 47). Efficient implementations of this optimization problem address the high number of k-mers, $|F|$. The problem is simplified by the use of equivalence classes (48) by grouping k-mers together into compatible sets. For a mammalian genome, the number of equivalence classes is orders of magnitude smaller than the number of possible k-mers. By applying multiple iterations of the expectation maximization procedure, α_t can be estimated (49) from the observed mapping of fragments F to the reference transcripts T. The likelihood function can be modified to take other factors such as GC content into account. The two pseudo-alignment algorithms share many optimization details while employing different ways of mapping k-mers to equivalence classes, and as such, they have similar but not the same quantification results. The majority of reads or k-mers map uniquely to a single transcript, resulting in a trivial identification of the transcript of origin. Ambiguous mapping affects only a subset of cases resulting in slightly different results by the two implementations.

RNA-Seq alignment and quantification performance

To benchmark existing RNA-Seq aligners, we first review a series of prior publications that benchmarked aligners. There are many aligners, and this makes it difficult to obtain consistent comparisons between all pairs of methods. We aggregated the results from the published benchmarking publications to identify the most commonly used implementations with the most promising performance. We focus on summarizing the results by comparing commonly used aligners (2, 4, 5, 47, 50–54). In the first benchmark (55), a set of alignment-based methods were compared by quality and runtime. In terms of speed, HISAT2 and STAR outperform the other tested alignment based methods by a significant margin. This benchmarking study found that default parameter settings have a significant impact on the precision and recall of the alignment algorithms. In a direct comparison, HISAT2 is sensitive to the choice of parameters and performs significantly worse than STAR with the default settings. For the benchmark, synthetic reads were created by selecting exon sequences and modifying them by introducing errors. Consequently, for low mismatch penalties, HISAT2 performed similarly to STAR. In another study (56), the authors showed that the fraction of aligned reads is affected by the read length. Short reads of 36 nucleotides were harder to align than longer reads. The sensitivity improves for longer reads. It was also shown that sequences of 70 nucleotides align with the same sensitivity as reads with a length of 300. Alignment-free RNA-Seq quantification methods show highly concordant RNA counts with expression profiles correlating around 0.98 between kallisto and Salmon. Especially, RNA segments from protein-coding regions show a high correlation. In another benchmarking study (21) alignment-based methods show high concordance with each other as do alignment-free methods. The authors conclude that for protein-coding genes, HISAT2, kallisto, and Salmon return similar results. The behavior of alignment-based methods diverges from that of alignment-free methods for ribosomal RNAs and tRNAs. In the case of tRNAs, alignment-free methods are shown to be outperformed by alignment-based methods. A comparison of gene expression profiles from alignment-free methods and alignment based methods show a Pearson correlation in a range of 0.68-0.72. Significant degradation of performance was observed for all methods for lowly expressed genes and genes of short length. While highly expressed genes with long transcripts are detected similarly with the two methods. Reducing the k-mer size from 31 to 21 improves tRNA detection in alignment-free methods. In most practical applications, the main objective is the identification of changes in gene expression between two conditions. Using transcript count or gene count as a benchmark of RNA-Seq alignment algorithm performance was shown to over-estimate the quality of alignment algorithms (57). They show that most tools struggle to detect differentially expressed genes accurately when tested on synthetic data even though the absolute gene expression seems quantified and replicable at high accuracy. As such, the authors of the same study recommend using differential gene expression as a benchmark for assessing alignment quality.

Multiple studies benchmarked the speed of collections of alignment tools (20, 56–62). As a general trend, the speed of the new aligners outperforms older implementations by order of magnitude while the alignment accuracy improved modestly. We decided to score the alignment tools based on these benchmarks manually (Table 1). The overall rank of the alignment algorithms is based on their performance in a set of benchmarks reported in eight publications. Across all benchmarks, a dominant algorithm with clear performance advan-

tage in all metrics does not seem to exist and depending on the test criteria, the performance and ranks mostly conflict. Also, as a general observation, benchmarks by the group that developed the algorithm commonly rank their own software high (63).

Based on the ranking shown in Table 1 and considerations of cost-effectiveness, we selected five top aligners for an in-depth follow-up benchmark. The main goal of our benchmark is to identify aligners that would be most suitable for large scale RNA-Seq processing projects. We first compared the runtime of the algorithms and how well they scale with additional computing threads. Our analysis demonstrates that kallisto outperforms all other implementations (Fig 1a). BWA takes the most amount of time to execute. From the alignment based methods, HISAT2 is the fastest, and it scales well with high thread count. STAR, which used to be the fastest aligner, uses significantly more memory (16GB). Increasing the thread count does not yield a linear improvement with a thread count in any of the tested alignment algorithms. All the aligners show significant diminishing return with an increase in the number of threads due to IO overhead.

A common application of RNA-seq studies is the identification of differentially expressed genes between a set of control samples and a set of perturbation samples. For benchmarking aligners with real data, we manually identified 80 publicly available studies containing RNA-seq data where differential gene expression analysis can be clearly applied. We compared the aligners performance by absolute expression quantification, differential gene expression, and downstream enrichment analysis. There is multiple suitable algorithms for specialized for differential gene expression analysis (64–68). In this study we limit the scope to a single algorithm (65). The correlation between 356 matched and unmatched RNA-Seq samples and the computed absolute expression show that the matched samples are highly correlated (Fig 1). kallisto and Salmon produce the most similar profiles to each other while STAR, HISAT2, and BWA also share similarities. The alignment-free and alignment based methods cluster separately from each other. On average HISAT2 profiles have the highest similarity to all other methods with a Spearman correlation coefficient of 0.814. Figure S3b and c show the number of aligned reads for each aligner (Fig S3b, Fig S3c). BWA tends to align the most reads, followed by HISAT2. STAR returns gene expression profiles with the lowest number of gene counts. The number of reads is consistent across most samples, with only some exceptions where STAR has a higher read count compared to HISAT2 (Figure S3b). To confirm the notion that alignment-free methods have higher error rates for genes with low expression compared to alignment based methods (69), we select two sets of genes, one representing genes with low expression (10% to 40% percentile range) and genes with high expression (70% to 100% percentile range). Each group contains 4,725 genes (Fig S1a). We do not observe a clear difference between the two percentile ranges comparing kallisto, Salmon, STAR, and HISAT2 (Fig S1b). The similarity in this comparison means that all algorithms perform similarly on low expressed genes and highly expressed genes. Only when comparing to BWA, there is an observable discordance, where lowly expressed genes show differences, and these differences are absent for highly expressed genes. As such, we cannot conclude that there is a systematic difference among the benchmarked aligners in regards to the quantification accuracy for genes with low expression. The unique behavior of the BWA algorithm is the only observed exception. To test how reproducible the alignment methods quantify gene level counts we compared sets of control samples from the differential expression studies. We tested the ingroup correlation between the control samples compared to the correlation against the other samples. No statistically significant difference between the methods was observed. Next, we calculated differential gene expression profiles using limma (65). This resulted in 80 differential gene expression profiles. While unmatched differential expression profiles do not show significant similarity, the matched profiles are correlated (Pearson correlation coefficient of 0.697 - 0.771) (Fig 1c). BWA is the outlier in the comparison to the other four algorithms following a similar pattern as seen in the absolute gene expression benchmark. We further wanted to investigate how these observed differences propagate into conclusions drawn from a downstream analysis. For the final benchmark, we performed enrichment analysis on the genes up-regulated upon perturbation. We performed enrichment analysis using Enrichr (70) on Gene Ontology biological processes. A biological process was significantly enriched after testing significant using the Fisher Exact test and a corrected p-value below 0.05 (Tab 2). BWA returns the lowest number of significant biological terms with 363. STAR identifies 988 gene sets as significantly enriched and this is suggestive that STAR is the most sensitive method to identify the "correct" levels of genes' mRNA. We then calculated the overlap of significant terms of alignment algorithm pairs. The found significant biological terms overlap significantly between the methods. Even though BWA identifies the most significantly differentially expressed genes it identifies the lowest number of significantly enriched biological terms (Fig S4c).

	score	benchmark count	avg score
Novoalign	6	2	3
HISAT2	8	3	2.67
Salmon	5	2	2.5
kallisto	10	5	2
Sailfish	4	2	2
Bowtie2	4	2	2
BWA	2	1	2
CLC	2	1	2
STAR	6	4	1.5
GSNAP	1	2	0.5
Tophat2	1	2	0.5
HISAT	0	2	0
OLEGO	0	1	0

Table 1. Averaged scores across eight review papers benchmarking the performance of alignment and pseudo-alignment tools.

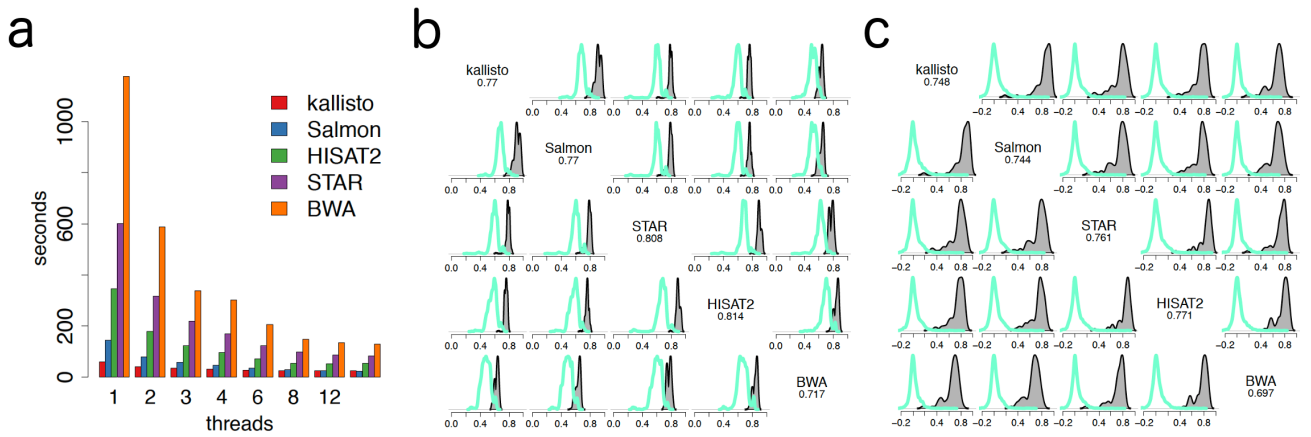

Fig. 1. Benchmark of a set of popular transcript abundance quantification methods. a) Required runtime in seconds for a different number of concurrent threads to align 3,161,833 reads to the human reference genome from SRR2972202. b) Pairwise distribution of Spearman correlation coefficients between different alignment algorithms for absolute expression quantification. Profiles were retrieved from 356 samples belonging to 80 manually selected experiments providing control and perturbation samples. The average correlation between matching (gray) and non-matching (green) samples as determined by different aligners. c) The distribution of pairwise Pearson correlation coefficients of differential expression profiles generated from the 80 datasets.

	kallisto	Salmon	STAR	HISAT2	BWA
kallisto	**647**	546	504	431	285
Salmon	546	**712**	565	442	288
STAR	504	565	**988**	527	283
HISAT2	431	442	527	**610**	298
BWA	285	288	283	298	**363**

Table 2. Number of significantly enriched biological terms from 80 differential gene expression profiles for the top 250 up-regulated genes computed with five transcript quantification tools following enrichment analysis in Enrichr for the libraries GO Biological Processes, KEGG Pathways, ChEA transcription factors, and Human Phenotypes. Off-diagonal values represent the number of overlapping significant biological terms.

k-mer length in pseudo-alignment

k-mers are the overlapping fragmentations of reads that are mapped to a position on the reference sequence. In the case of transcript quantification, these k-mers are mapped to a transcript by the use of a hash table or other fast index data structure. The length of a k-mer is predefined during the process of building the index to perform the lookup efficiently. Very short k-mers can map to many locations on the genome. A random sequence of the four possible nucleotides A, T, G, and C of length ten would amount to 4^{10} unique possible sequences. In comparison to a mammalian genome with a length of $\approx 3^{20}$ we expect to match a k-mer of length 10 to more than 3000 positions assuming a uniform random distribution of nucleotides. By increasing the length of k-mers the sequences will become distinctly more unique. In practice DNA sequences are more restrictive than random sequences. With a k-mer length of 31 building an index in kallisto on a mouse reference results in 734,746 contigs and 100,614,952 unique k-mers. A k-mer size of 15 results in 46,244,720 contigs and contains 76,010,263 k-mers. We tested the performance by changing the indexes in kallisto and Salmon based on those two settings for the dataset GSE121539 containing 14 paired-end read of mouse samples derived from pancreatic islets (71). The percentage of aligned reads using a k-mer length of 31 results in a significantly higher number of reads mapped to contigs. On average, 39.6% more reads are mapped compared to a k-mer length of 15. Additionally, the algorithm fails to accurately estimate the average fragment length for the short k-mer index. The estimated fragment length is between 16 and 25. For a k-mer setting of 31, kallisto predicts a much more reasonable number 250bp. The resulting gene expression profiles for the two settings are still nearly identical with a mean Spearman correlation of 0.941. By inspecting the individual genes showing the most significant decreases in gene count are the vomeronasal genes as well as olfactory receptors comprising two large gene families. The reduced gene count for these gene families is most likely due to high sequence similarity between the genes, interfering with the accurate estimation of transcript counts since k-mers are not directly mapped to unique genes. Modifying the k-mer length for Salmon did not change quantification results as strongly as in kallisto. For Salmon a k-mer length of 31 results in only 2% more aligned reads compared to a k-mer size of 15. The calculated gene expression profiles of kallisto using 31 k-mers lengths are more similar to the gene expression profiles from Salmon using 15 k-mers (mean Spearman correlation of 0.948) compared to kallisto with k-mers of length 15 (mean Spearman correlation of 0.941). This suggests sensitivity of kallisto to k-mer length and a degradation of transcript quantification for short k-mer length.

RNA-seq concordance to qPCR

In a study RNA-Seq derived fold changes of gene expression were compared to wet-lab validated qPCR assays for 18,080 protein-coding genes (60). After removing outliers from the qPCR data we are left with more than 4800 genes for which we also have RNA-seq fold change information

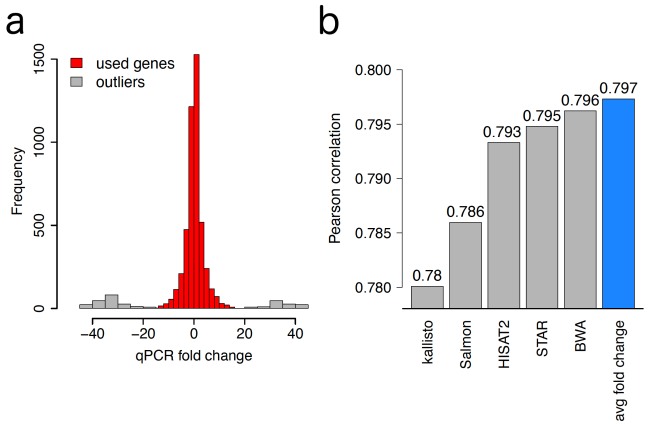

Fig. 2. Comparison of RNA-Seq data to wet-lab validated qPCR for 18,080 protein coding genes. a) Reported fold change derived by qPCR. Three sub-populations of genes identified by k-means (3 clusters). In red is the gene population used for benchmarking RNA-seq concordance. b) Performance of alignment algorithms measured by concordance to qPCR. The concordance, by taking the average fold change of all five methods, is shown in blue.

2a. All alignment algorithms have a high Pearson correlation when compared to 2b. When combining the results from all five alignment algorithms the correlation to qPCR is highest. The alignment free methods kallisto and Salmon have the lowest similarity. The absolute differences of the algorithms is similar.

Data reusability and API documentation

The findability, accessibility, interoperability, and reusability (FAIR) guiding principles are part of a global effort toward making biomedical research data more impactful (72). The challenge of building the necessary infrastructure for seamless integration of existing data into new workflows can be overcome by following standards that are endorsed by the community, which includes academia, industry and funding agencies. The need for data management standards is motivated by the goal of enhancing the knowledge about biological processes through integration and reuse of data by the community after the data have been published. The current state at which data is stored and published stands in the way of rapid and efficient reuse. The principles to enhance data sharing and reuse are grouped into four categories related to best practice of data stewardship. These categories are concerned with data being FAIR. The principles are deeply related to the increasing connectivity of computer networks, allowing instantaneous information exchange.

At the core of the FAIR principles, if data producers were to fully annotate their metadata and data using semantics, which is understandable by machines, digital resources would become FAIR. Adhering to these principles, it is expected that data reuse will become easier with the help of automation. While FAIR is primarily focused on metadata, it is also applicable to a broader set of digital resources such as to tools, APIs, workflows, and more.

Taking advantage of the interconnected architectural style of the web, Representational State Transfer (REST) promises to be a solution to transform the current state of silos of data into an integrated web of data (73). An essential prerequisite for the exchange of such information is an infrastructure of interconnected services that expose their functionality to the outside world. This interconnection can be achieved through well defined APIs. With the use of JSON validation schemas (74), digital, interconnected data processing ecosystems can be created (75). With increasing networking bandwidth, data and computational services can co-exist in a decentralized infrastructure, that is unbound to specific hardware and locale, allowing resource scaling and replication (76).

The OpenAPI, formerly Swagger standard, is a specification for allowing RESTful APIs to be efficiently described, produced and consumed by other services (77–79). To enable such a process, the files describing the APIs are machine readable. The OpenAPI specifications can be generated in YAML Ain't Markup Language (YAML) or JavaScript Object Notation (JSON) (80). Specialized editors, such as the Swagger editor, can be used to construct the description files manually. A more efficient way to generate the OpenAPI description files is by adding OpenAPI documentation directly into the corresponding code. The embedding of OpenAPI documentation ensures that when changes are made at the code level, the API description can also be updated at the same time and place, resulting in better adherence to consistency. For example, if a specific endpoint of an API is removed, the OpenAPI documentation would be removed in the same step. When rebuilding the project, the autogenerated YAML and JSON description files would automatically have the API endpoint removed as well. APIs that are annotated by the OpenAPI standard can be registered at Swagger.io. The SmartAPI registry (81) is an extension of the OpenAPI standard. Repositories of SmartAPI compliant APIs from the life science domain can be found at the BioCatalogue (82). The APIs in BioCatalogue ranges from text mining to enzyme kinetics, covering a broad spectrum of functionality. Other examples of API driven tools are the Protein API (83) and the STRING DB API (84).

Docker encapsulated API driven microservices

Docker is a widely used virtualization technique. It allows the generation and deployment of workflows with all its dependencies included in a binary file (85). A Docker container encapsulates a operating system, programs such as python and R distributions as well as packages dependencies and scripts (86, 87). By Upload of generated Docker images on a centralized online repository called Docker Hub provides efficient means of distributing reproducible work environments. Most Dockerized pipelines are means for programmatic use (88–91), but there has been some effort in supporting graphical user interfaces (92). An example of a working ecosystem that increases data interoperability is the connection of the individual micro and macro services provided by ARCHS4, Elysium, and BioJupies (20, 93). All of these services are standalone web-services that can be used independently. The basic structure shows how such services can

work together while being hosted on completely independent hardware and locations communicating via APIs.
There are two types of data being considered in this framework. Publicly available raw reads from the SRA database that are annotated by the GEO, and privately held datasets in either raw sequence format or previously aligned.

Elysium. The Elysium APIs are cloud-based and built for scalability (94). To achieve this, critical aspects of the pipeline are averting bottlenecks, for example, relying on individual servers for handling heavy duty tasks such as file upload and data processing. The system relies on control servers that upon request produce temporary user credentials for allowing direct interaction with the Amazon S3 bucket for file upload and submission of job instructions to the compute backend. The system is composed in a modular fashion with modules for upload, sequence job scheduling, RNA-Seq alignment, and a graphical user interface. The file upload server is hosting an API to facilitate file upload directly to the Amazon S3 bucket without passing data through an intermediary instance. In order to upload a file, a user needs to establish a user account. With this account, encrypted tokes can be requested. POST requests are sent directly to Amazon S3 without the need for Amazon credentials. To manage the permissions of users to upload data, a user can request an encrypted token which allows data uploads for a limited amount of time. The RNA-Seq data is stored in a private repository. The job scheduler API supports submissions of alignment jobs for files already uploaded to the cloud repository. The user can specify a single, or paired-end, read files, and the reference genome for the alignment. The submitted jobs are moved into a processing queue that serves the job based on its description on a first-come-first-serve basis. The job scheduler also handles instance scaling of the processing backend. It can be configured with a minimum, and a maximum number of active compute instances, as well as a scaling factor relative to the number of queued alignment jobs. The alignment container is a pre-configured Docker container that deploys automatically into the elastic Amazon cluster. The container runs kallisto (52) as the primary alignment tool. The alignment tool performs transcript quantification and gene count transformation and then moves the alignment results to the user workspace. After a job completes, the alignment container will request a new job from the job scheduler API. The Elysium GUI is a light-weight graphical user-interface that is serving the underlying API functionality (https://amp.pharm.mssm.edu/elysium). It supports the generation of user accounts, file upload, job generation, and submission. Elysium displays job processes and results in the user workspace with download links for transcript level quantification and gene level count processed data files. Since going online, more than 10,000 samples have been uploaded to Elysium, mainly via the BioJupies interface. The distribution of compute-time and cost generated for the uploaded samples was measured for a period of several weeks shows that most FASTQ samples can be computed in less than 10 minutes and at a cost below 1(Fig S4a). The need for an elastic compute environment that can adapt to real-time workloads is demonstrated by the cost of alignment for 10,000 samples which is below $100.

BioJupies. The BioJupies service (93) enables users to generate customized, interactive Jupyter notebooks containing analyses of RNA-Seq data through an intuitive user interface. The notebooks contain executable Python and R code, interactive visualizations, and rich annotations that provide detailed explanations of the results. The notebooks also provide details and references about the methods used to perform the analyses. The notebooks are made available to users through a permanent and shareable URL. The automatically generated notebooks can be downloaded and modified on the user's local computer through the execution of a Docker image that contains all the data, tools, and source code needed to rerun analyses. Currently, the BioJupies plug-in library consists of 14 computational plug-ins subdivided into four categories: exploratory data analysis, differential expression analysis, enrichment analysis, and small molecule query. These plug-ins analyze the data and display interactive visualizations such as three-dimensional scatterplots, bar charts, and clustered heatmaps with enrichment analysis features. The modular design of BioJupies provides the mechanism needed for other bioinformatics tool developers to integrate their tools. By integration of ARCHS4 data, existing RNA-seq gene expression studies have been pre-analyzed and can be readily loaded as a Jupyter notebook. Through the Elysium API, users can upload FASTQ files for alignment.

Use-cases. We highlight three data flows, depicted as dashed lines, as an example of the flexibility of the use of micro-services to build a modular solution for a variety of data analysis workflows.

Use Case I A user can perform an interactive exploratory analysis of previously published gene expression data through the ARCHS4 web-interface. Publicly available experiments that are shared on GEO and are annotated to have RNA-Seq data are periodically added to the ARCHS4 repository. When new RNA-Seq samples are identified, a new alignment task is added to the Elysium processing queue. As a prerequisite, data has to be reprocessed from a raw sequencing file deposited in the SRA database. The raw data is processed in the cloud alignment infrastructure encapsulated in the Elysium cloud alignment module.

Use Case II A user has raw sequencing data from a set of RNA-Seq experiments. The user wants to identify similar gene expression profiles previously published. After identifying similar experiments, the user wants to build a larger gene expression matrix to model context-specific gene correlation patterns.
The challenges are to identify similar samples. Without large scale pre-processed gene expression repositories, the only way to identify relevant experiments is by performing a metadata search on GEO. The search will return a set of potential candidate datasets that might be of interest. The user will then have to download all raw sequencing samples and process them in the same way. After the homogeneous process-

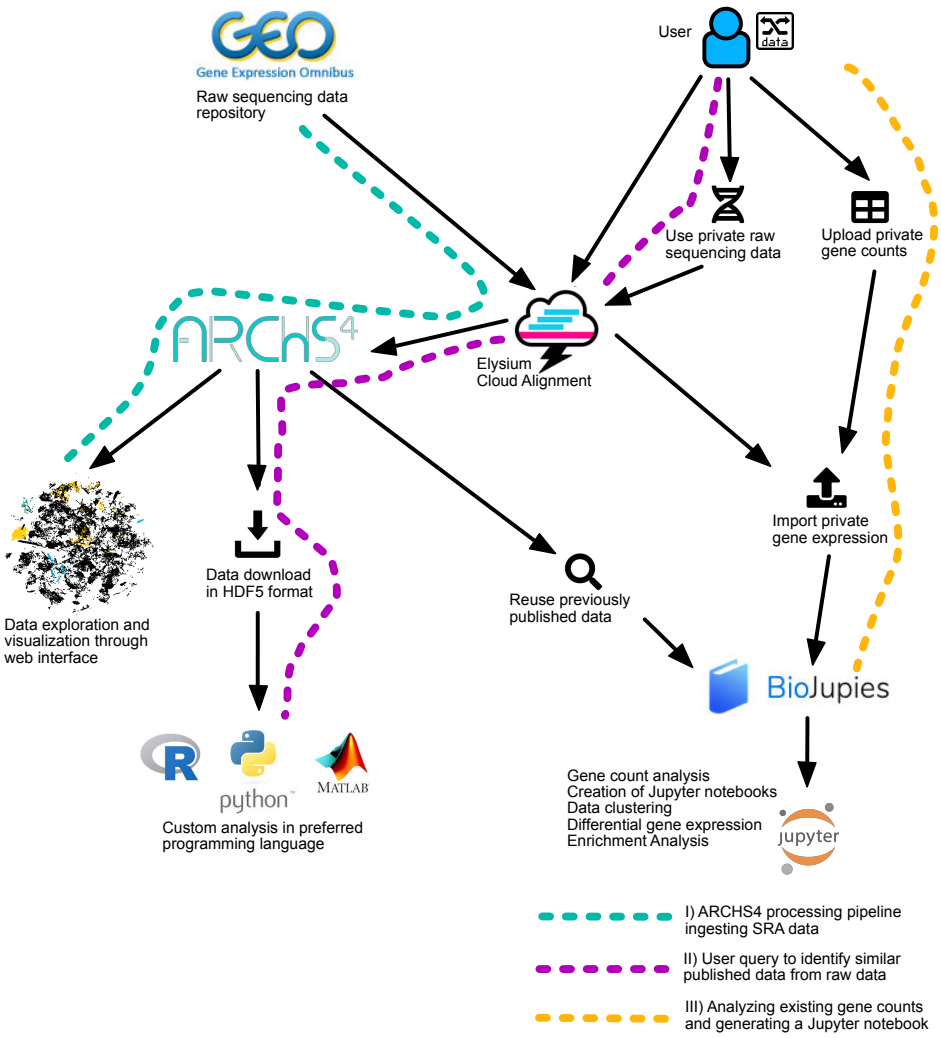

Fig. 3. The gene expression reuse ecosystem. The diagram shows how data flows through the micro-services. Three highlighted data flows labeled I to III are explained in more detail to showcase typical use cases.

ing, the data can be compared on the data level. At this stage, it can be decided whether the data contains useful additional information or not.

This example highlights the complexity of integrating existing data into a new research project. The dependency of homogeneous processing of the source data adds workload to the workflow. Since search can only be performed at the metadata level, there is no guarantee that the additional data can add value to the project. As a result, most studies only rely on the data that was generated by the original study.

The pre-processing of RNA-seq and the availability of the same processing pipeline to anyone enables users to reduce the burden of identifying relevant datasets that are publicly available. First, the raw RNA-seq reads are uploaded to an online repository through the Elysium interface utilizing the a cloud storage microservice. Then an alignment job can be requested by the cloud alignment control service. Once the data is processed, the Elysium interface supports a reference search in the ARCHS4 dataset. This search redirects the user to the ARCHS4 user interface with similar samples highlighted in the sample overview. The data can be downloaded in a prepackaged file including an R dataframe that contains the matrix ready for integrative analysis..

Use Case III In this example, a user has aligned RNA-Seq data using a custom aligner on a local machine. The data can be uploaded to the BioJupies system in gene count format. In order to perform differential gene expression analysis the user groups the samples manually into two sets using the BioJupies interface. Besides the standard exploratory analysis of the data such as PCA plots and library size analysis, differential gene expression analysis will be performed automatically and loaded into a Jupyter notebook. The notebooks embed the code of the data analysis next to plots and analysis results.

Methods

A. Transcript quantification benchmark. All samples are from Homo sapiens RNA-Seq and were aligned against release-96 Ensembl GRCh38. Software versions used were STAR 2.7.1a, BWA 0.7.17-r1188, kallisto 0.45.1, Salmon v0.13.1, HISAT2 version 2.1.0 and featureCount Version

1.6.4. We manually selected 80 datasets consisting of a total of 356 control and treatment samples. The datasets and their samples are listed in Tab **??**. They consist of drug and gene perturbations as well as disease vs. normal tissue comparisons.

To compare the performance of the 5 transcript quantification methods, we first manually scored aligners based on review articles that benchmarked aligners. We focus on implementations that can run efficiently with limited time and memory complexity. The primary use case is a cost-efficient solution with good quality of the alignment. In our literature review, if an algorithm was benchmarked as the best among all tested alternatives, we gave it 3 points. If the algorithm came in at 2nd place, it received two points, and when it is 3rd, it will received one point. We calculated the final score as the average of scores an algorithm received across all reviews. Novoalign is ranked at the top based on 2 studies. However, it is not included in the following comparison of our benchmarking because it is significantly slower than all other alternatives. To benchmark paired-end read alignment for k-mers of length 15 and 31 in kallisto and Salmon we used the mouse RNA-Seq data published in (71). We created a Docker image containing all scripts to build the reference index and run alignment on files in the SRA database. All dependencies are pre-installed using Conda. The Docker image is freely available in Docker Hub (maayanlab/alignmentbenchmark). We ran all benchmarks on AWS EC2 c5.4xlarge instances supporting 16 threads and 32GB of memory. Except for BWA, these instances are sufficient to build the index file for transcript quantification. BWA needs close to 50GB for mammalian genomes. We deposited prebuilt genome index files for human and mouse Ensembl v96 on publicly available S3 buckets. A script to download human and mouse index files with the corresponding links to the files is provided in the Docker container. Gene counts are computed by adding all transcript counts belonging to the same gene rounded to the nearest integer for kallisto and Salmon. For STAR we used the "–quantMode GeneCounts" option. For HISAT2 and BWA, we used featureCounts from the Subread package. We calculate differential gene expression for each differential gene expression dataset and each transcript quantification algorithm using the R limma package v3.6 (65). To compare similarity between the gene counts, we use Spearman correlation, for differential gene expression profiles, we calculate Pearson correlation. We apply geneset enrichment using the enrichR R package for the Enrichr geneset libraries GO Biological Process 2018, KEGG 2019 Human, ChEA 2016, and Human Phenotype Ontology. We identify significantly overrepresented genesets after multiple hypothesis testing correction using the Bonferroni method and a p-value of less than 0.05.

B. Cloud based transcript quantification. The file upload and job scheduler APIs are dockerized tornado webservers. The Docker containers are fully configurable through passing environment parameters during the Docker container launch. User credentials and job descriptions are stored in a MySQL database shared between the file upload server and the job scheduler. The AWSalignment container is a dockerized Ubuntu server supporting python and R with a MySQL client. AWSalignment is pre-configured with kallisto 0.44 for sequence alignment. For the transfer of transcript quantification results back to the user's personal account on Amazon S3, the python boto library is used. Transcript to gene mapping is performed with an R-script and the annotation from the cDNA annotation file from Ensembl. The annotation version is Ensembl 90, and the reference genome for human and mouse are GRCh38 and GRCm38, respectively. These files are downloaded from the Ensembl website. The Elysium graphical user interface is implemented as a single HTML file. This is achieved with support from the JQuery, Bootstrap, and CSS Grid libraries. The page is hosted via the job scheduler tornado server. APIs are registered and documented with SwaggerHub and SmartAPI.

C. Approximate nearest neighbor search in high dimensional space. In order to efficiently compute the similarity of the gene expression profile from Elysium to the complete set of ARCHS4 gene expression profiles, we apply a metric embedding to project the original data into a lower dimensional space. The key objective is to avoid the distortion of the pairwise distances by the projection and reducing the dimensionality of our data to perform fast nearest neighbor calculations. For a mapping Φ, we aim to minimize a relative distortion that can be defined as:

$$\max_{x_1,x_2 \in U} \frac{|d_U(x_1,x_2) - d_V(\Phi(x_1),\Phi(x_2))|}{d_U(x_1,x_2)}$$

U and V are metric spaces. U is the original space while V is the new metric space into which we want to project the data. In order to achieve performance gains, we wish the dimensionality k to be significantly smaller than that of U. A transformation that can achieve such a dimensionality reduction is the Johnson Lindenstrauss transformation (JLT). The standard definition of a JLT guarantees to limit the distortion when the following three conditions are met:

- Spherical symmetry: For any orthogonal matrix A $\in$ $O(d)$,ΦA and Φ have the same distribution
- Orthogonality: The rows of Φ are orthogonal to each other
- Normality: The rows of Φ are unit length vectors

A simplification to JLT was previously introduced (95) allowing the JL error to be preserved when entries of Φ are independently chosen using the normal distribution $N(0, 1/d)$. If the rows of Φ contain sufficient entries, the normality and orthogonality are approximately preserved. The assumptions simplify the generation of Φ significantly. When the above prerequisites are met, the error induced by a JLT will stay small. However, the assumptions of the original transformation are not completely satisfied when applied to gene expression. For the application of Elysium, we require the nearest neighbor ranking of a gene expression profile to be conserved. We tested the rank conservation of gene expression

signatures given multiple transformations Φ_k with k between 10 to $5,000$.

Conclusion

Here we reviewed and benchmarked existing transcript quantification tools specifically with the purpose of cost efficiency and quantification accuracy. Most modern aligners are suitable for the alignment of RNA-Seq reads for projects involving hundreds of samples. For large scale re-sequencing efforts such as recount2 and ARCHS4, the performance of alignment algorithms becomes a significant fiscal hurdle when operating on a limited budget. With nearly 400,000 publicly available RNA-Seq samples currently available (June 2019) even a speed improvement of a factor of two can make a difference. While there have been numerous benchmarks of existing tools, they are not always in agreement. Here we extended the existing set of benchmarks to several new approaches applied to real publicly available datasets. We identified 80 publicly available RNA-Seq studies analyzed by a complete workflow from aligning the raw reads, to differential expression, to enrichment analysis. We found significant speed and cost improvements for alignment-free algorithms without detecting systematic differences in quality when comparing to alignment based algorithms such as STAR or HISAT2. We would recommend kallisto or Salmon for applications where cost plays a critical role. Since there are always concerns about alignment-free methods, HISAT2 is the best alternative when it comes to speed and memory footprint. HISAT2 performance in speed beats STAR by a factor of 1.7 and BWA by a factor of 3.4. Even then, kallisto is a factor of 5.7 faster than HISAT2. While BWA detects the highest number of differential expressed genes and results in the highest gene counts, it seems to have a reduced sensitivity in detecting meaningful biological modules as shown by the lowest number of enriched biological terms from enrichment analyses. STAR seems to be the most sensitive algorithm in terms of identifying relevant pathways and biological processes. When analyzing the effect of k-mer length, we identified a potential drawback with the ability of kallisto to accurately predict paired-end reads and a significant decrease of mappable reads to the transcriptome for k-mers of length 15. We recommend keeping the default parameter settings when using kallisto. In light of previously published conflicting benchmark results, we developed a Docker image of our alignment workbench that is completely pre-configured with all dependencies to run the benchmark for the five aligners. The major components of the benchmark are illustrated in Jupyter notebooks that can be opened and run directly from the Docker image. Modifications of these notebooks and scripts should enable the extension of this application to benchmark datasets and alignment tools in the future. The Jupyter notebook encompasses index generation, index upload to the cloud, index download from the cloud, alignment of SRA files from a file, runtime benchmarks, and the aggregation of gene counts of all aligners into a single combined file for efficient post-processing. The manually curated set of differential gene expression profiles can be a useful benchmark in the future analyses of other alignment scenarios. Reproducibility and data compatibility is at the core of large re-sequencing efforts. We introduced some essential elements for the development of web-based API driven microservices that can be combined to create complex workflows around RNA-Seq data. We show the integration of these concepts withe example workflows spanning Elysium, BioJupies, and ARCHS4. Through its API, Elysium has native programmatic access to its functionality. Through automatic instance scaling of the compute back-end, the maintenance and operating costs of the cloud alignment jobs is kept low. The average cost for processing a FASTQ file is currently below ¢1 as previously reported (20). A critical capability of Elysium is the leveraging of many of the available features of the cloud infrastructure, and as a consequence minimizing possible bottlenecks in data transfer and computational load. Upload speed per file is currently at 60MB/s, and this speed is independent of concurrent users. The dynamic cluster size and fast load times of Docker containers result in a scaling response time of fully functional resources in less than 5 minutes. The fast upstart times of new instances keep wait times from submission to final results on average below 15 minutes, and this waiting time is independent of traffic. Applications other than RNA-Seq analysis could be incorporated into the existing Elysium framework with the addition of other types of specialized Docker containers. The microservices and APIs presented here, in combination with the GUI, increase interoperability of existing RNA-Seq data analysis pipelines, facilitating users with minimal computational skills with the ability to process their own data using the same pipeline implemented for creating the ARCHS4 resource. The uniform processing can place the newly processed data in the context of more than 370,000 previously published RNA-Seq data-sets currently available at the ARCHS4 resource. As of June 2019, ARCHS4 and BioJupies enabled data analysis for 35,000 unique users in a time span of less that two years. This demonstrates the utility and need for developing scalable user friendly RNA-Seq bioinformatics resources.

ACKNOWLEDGEMENTS

This work is partially supported by the National Institutes of Health (NIH) grants U54-HL127624 (LINCS-DCIC), U24-CA224260 (IDG-KMC), and OT3-OD025467 (NIH Data Commons), as well as cloud credits from the NIH BD2K Commons Cloud Credit Pilot project to AM.

Bibliography

1. Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.
2. Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357, 2012.
3. Ruiqiang Li, Chang Yu, Yingrui Li, Tak-Wah Lam, Siu-Ming Yiu, Karsten Kristiansen, and Jun Wang. Soap2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25 (15):1966–1967, 2009.
4. Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.
5. Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *bioinformatics*, 25(14):1754–1760, 2009.
6. Chi-Man Liu, Thomas Wong, Edward Wu, Ruibang Luo, Siu-Ming Yiu, Yingrui Li, Bingqiang Wang, Chang Yu, Xiaowen Chu, Kaiyong Zhao, et al. Soap3: ultra-fast gpu-based parallel alignment tool for short reads. *Bioinformatics*, 28(6):878–879, 2012.
7. Peter Amstutz, Michael R Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael

Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, et al. Common workflow language, v1. 0. 2016.
8. Kate Voss, Geraldine Van der Auwera, and Jeff Gentry. Full-stack genomics pipelining with gatk4+ wdl+ cromwell. *F1000Research*, 6, 2017.
9. John Vivian, Arjun Arkal Rao, Frank Austin Nothaft, Christopher Ketchum, Joel Armstrong, Adam Novak, Jacob Pfeil, Jake Narkizian, Alden D Deran, Audrey Musselman-Brown, et al. Toil enables reproducible, open source, big biomedical data analyses. *Nature biotechnology*, 35(4):314, 2017.
10. Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo Del Angel, Manuel A Rivas, Matt Hanna, et al. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature genetics*, 43(5):491, 2011.
11. Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86, 2010.
12. Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, 41(W1):W557–W561, 2013.
13. Andreas Sundquist. Dnanexus free account: next-gen sequence analysis in the cloud. *SEQanswers, Apr*, 27, 2010.
14. Gaurav Kaushik and Brandi Davis-Dusenbery. Building portable and reproducible cancer informatics workflows: An rna sequencing case study. In *Cancer Bioinformatics*, pages 39–64. Springer, 2019.
15. Christophe Van Neste, Yannick Gansemans, Dieter De Coninck, David Van Hoofstat, Wim Van Criekinge, Dieter Deforce, and Filip Van Nieuwerburgh. Forensic massively parallel sequencing data analysis tool: Implementation of myflq as a standalone web-and illumina basespace®-application. *Forensic Science International: Genetics*, 15:2–7, 2015.
16. Jeremy Leipzig. A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, 18(3):530–536, 2017.
17. Matthew N McCall, Harris A Jaffee, Susan J Zelisko, Neeraj Sinha, Guido Hooiveld, Rafael A Irizarry, and Michael J Zilliox. The gene expression barcode 3.0: improved data processing and mining tools. *Nucleic acids research*, 42(D1):D938–D943, 2013.
18. Leonardo Collado-Torres, Abhinav Nellore, Kai Kammers, Shannon E Ellis, Margaret A Taub, Kasper D Hansen, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. Reproducible rna-seq analysis using recount2. *Nature Biotechnology*, 35(4):319–321, 2017.
19. Naim Al Mahi, Mehdi Fazel Najafabadi, Marcin Pilarczyk, Michal Kouril, and Mario Medvedovic. Grein: An interactive web platform for re-analyzing geo rna-seq data. *Scientific reports*, 9(1):7580, 2019.
20. Alexander Lachmann, Denis Torre, Alexandra B Keenan, Kathleen M Jagodnik, Hoyjin J Lee, Lily Wang, Moshe C Silverstein, and Avi Ma'ayan. Massive mining of publicly available rna-seq data from human and mouse. *Nature communications*, 9(1):1366, 2018.
21. Douglas C Wu, Jun Yao, Kevin S Ho, Alan M Lambowitz, and Claus O Wilke. Limitations of alignment-free tools in total rna-seq quantification. *BMC genomics*, 19(1):510, 2018.
22. Joshua Z Levin, Moran Yassour, Xian Adiconis, Chad Nusbaum, Dawn Anne Thompson, Nir Friedman, Andreas Gnirke, and Aviv Regev. Comprehensive comparative analysis of strand-specific rna sequencing methods. *Nature methods*, 7(9):709, 2010.
23. Ugrappa Nagalakshmi, Karl Waern, and Michael Snyder. Rna-seq: a method for comprehensive transcriptome analysis. *Current protocols in molecular biology*, 89(1):4–11, 2010.
24. Cristian Del Fabbro, Simone Scalabrin, Michele Morgante, and Federico M Giorgi. An extensive evaluation of read trimming effects on illumina ngs data analysis. *PloS one*, 8(12): e85024, 2013.
25. Yong Kong. Btrim: a fast, lightweight adapter and quality trimming program for next-generation sequencing technologies. *Genomics*, 98(2):152–153, 2011.
26. Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet. journal*, 17(1):10–12, 2011.
27. Mikkel Schubert, Stinus Lindgreen, and Ludovic Orlando. Adapterremoval v2: rapid adapter trimming, identification, and read merging. *BMC research notes*, 9(1):88, 2016.
28. Hongshan Jiang, Rong Lei, Shou-Wei Ding, and Shuifang Zhu. Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads. *BMC bioinformatics*, 15 (1):182, 2014.
29. Marc Sturm, Christopher Schroeder, and Peter Bauer. Seqpurge: highly-sensitive adapter trimming for paired-end ngs data. *BMC bioinformatics*, 17(1):208, 2016.
30. Yun-Lung Li, Jui-Cheng Weng, Chiung-Chih Hsiao, Min-Te Chou, Chin-Wen Tseng, and Jui-Hung Hung. Peat: an intelligent and efficient paired-end sequencing adapter trimming algorithm. In *BMC bioinformatics*, volume 16, page S2. BioMed Central, 2015.
31. Nicola Prezza, Francesco Vezzi, Max Käller, and Alberto Policriti. Fast, accurate, and lightweight analysis of bs-treated reads with erne 2. *BMC bioinformatics*, 17(4):69, 2016.
32. Assaf Gordon, GJ Hannon, et al. Fastx-toolkit. *FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit*, 5, 2010.
33. Robert Schmieder and Robert Edwards. Quality control and preprocessing of metagenomic datasets. *Bioinformatics*, 27(6):863–864, 2011.
34. Murray P Cox, Daniel A Peterson, and Patrick J Biggs. Solexaqa: At-a-glance quality assessment of illumina second-generation sequencing data. *BMC bioinformatics*, 11(1):485, 2010.
35. Linnéa Smeds and Axel Künstner. Condetri-a content dependent read trimmer for illumina data. *PloS one*, 6(10):e26314, 2011.
36. NA Joshi and JN Fass. Sickle: A sliding-window, adaptive, quality-based trimming tool for fastq files (version 1.33)[software], 2011.
37. Anthony M Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, 30(15):2114–2120, 2014.
38. Matthew D MacManes and Michael B Eisen. Improving transcriptome assembly through error correction of high-throughput sequence reads. *PeerJ*, 1:e113, 2013.
39. Matthew D MacManes. On the optimal trimming of high-throughput mrna sequence data. *Frontiers in Genetics*, 5:13, 2014.
40. GTEx Consortium et al. The genotype-tissue expression (gtex) pilot analysis: Multitissue gene regulation in humans. *Science*, 348(6235):648–660, 2015.
41. Geraldine A Van der Auwera, Mauricio O Carneiro, Christopher Hartl, Ryan Poplin, Guillermo Del Angel, Ami Levy-Moonshine, Tadeusz Jordan, Khalid Shakir, David Roazen, Joel Thibault, et al. From fastq data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. *Current protocols in bioinformatics*, 43(1):11–10, 2013.
42. Edward M McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM (JACM)*, 23(2):262–272, 1976.
43. Donald Adjeroh, Timothy Bell, and Amar Mukherjee. *The Burrows-Wheeler Transform:: Data Compression, Suffix Arrays, and Pattern Matching*. Springer Science & Business Media, 2008.
44. Juha Kärkkäinen and Peter Sanders. Simple linear work suffix array construction. In *International Colloquium on Automata, Languages, and Programming*, pages 943–955. Springer, 2003.
45. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography Conference*, pages 145–166. Springer, 2006.
46. Nicolas Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal rna-seq quantification. *arXiv preprint arXiv:1505.02710*, 2015.
47. Rob Patro, Geet Duggal, and Carl Kingsford. Salmon: accurate, versatile and ultrafast quantification from rna-seq data using lightweight-alignment. *Biorxiv*, 9:021592, 2015.
48. Marius Nicolae, Serghei Mangul, Ion I Măndoiu, and Alex Zelikovsky. Estimation of alternative splicing isoform frequencies from rna-seq data. *Algorithms for molecular biology*, 6(1): 9, 2011.
49. Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.
50. Thomas D Wu and Serban Nacu. Fast and snp-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881, 2010.
51. Daehwan Kim, Ben Langmead, and Steven L Salzberg. Hisat: a fast spliced aligner with low memory requirements. *Nature methods*, 12(4):357, 2015.
52. Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, 34(5):525, 2016.
53. Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from rna-seq reads using lightweight algorithms. *Nature biotechnology*, 32 (5):462, 2014.
54. Jie Wu, Olga Anczukow, Adrian R Krainer, Michael Q Zhang, and Chaolin Zhang. Olego: fast and sensitive mapping of spliced mrna-seq reads using small seeds. *Nucleic acids research*, 41(10):5149–5163, 2013.
55. Giacomo Baruzzo, Katharina E Hayer, Eun Ji Kim, Barbara Di Camillo, Garret A FitzGerald, and Gregory R Grant. Simulation-based comprehensive benchmarking of rna-seq aligners. *Nature methods*, 14(2):135, 2017.
56. Subazini Thankaswamy-Kosalai, Partho Sen, and Intawat Nookaew. Evaluation and assessment of read-mapping by multiple next-generation sequencing aligners based on genome-wide characteristics. *Genomics*, 109(3-4):186–191, 2017.
57. Mingxiang Teng, Michael I Love, Carrie A Davis, Sarah Djebali, Alexander Dobin, Brenton R Graveley, Sheng Li, Christopher E Mason, Sara Olson, Dmitri Pervouchine, et al. A benchmark for rna-seq quantification pipelines. *Genome biology*, 17(1):74, 2016.
58. Gregory R Grant, Michael H Farkas, Angel D Pizarro, Nicholas F Lahens, Jonathan Schug, Brian P Brunk, Christian J Stoeckert, John B Hogenesch, and Eric A Pierce. Comparative analysis of rna-seq alignment algorithms and the rna-seq unified mapper (rum). *Bioinformatics*, 27(18):2518–2528, 2011.
59. Lichun Jiang, Felix Schlesinger, Carrie A Davis, Yu Zhang, Renhua Li, Marc Salit, Thomas R Gingeras, and Brian Oliver. Synthetic spike-in standards for rna-seq experiments. *Genome research*, 21(9):1543–1551, 2011.
60. Celine Everaert, Manuel Luypaert, Jesper LV Maag, Quek Xiu Cheng, Marcel E Dinger, Jan Hellemans, and Pieter Mestdagh. Benchmarking of rna-sequencing analysis workflows using whole-transcriptome rt-qpcr expression data. *Scientific reports*, 7(1):1559, 2017.
61. Franck Rapaport, Raya Khanin, Yupu Liang, Mono Pirun, Azra Krek, Paul Zumbo, Christopher E Mason, Nicholas D Socci, and Doron Betel. Comprehensive evaluation of differential gene expression analysis methods for rna-seq data. *Genome biology*, 14(9):3158, 2013.
62. Sara Ballouz, Alexander Dobin, Thomas R Gingeras, and Jesse Gillis. The fractured landscape of rna-seq alignment: the default in our stars. *Nucleic acids research*, 46(10):5125–5138, 2018.
63. Paul C Boutros, Adam A Margolin, Joshua M Stuart, Andrea Califano, and Gustavo Stolovitzky. Toward better benchmarking: challenge-based methods assessment in cancer genomics. *Genome biology*, 15(9):462, 2014.
64. Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1): 139–140, 2010.
65. Matthew E Ritchie, Belinda Phipson, Di Wu, Yifang Hu, Charity W Law, Wei Shi, and Gordon K Smyth. limma powers differential expression analyses for rna-sequencing and microarray studies. *Nucleic acids research*, 43(7):e47–e47, 2015.
66. Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550, 2014.
67. Alyssa C Frazee, Geo Pertea, Andrew E Jaffe, Ben Langmead, Steven L Salzberg, and Jeffrey T Leek. Ballgown bridges the gap between transcriptome assembly and expression analysis. *Nature biotechnology*, 33(3):243, 2015.
68. Harold Pimentel, Nicolas L Bray, Suzette Puente, Páll Melsted, and Lior Pachter. Differential analysis of rna-seq incorporating quantification uncertainty. *Nature methods*, 14(7):687, 2017.
69. Leming Shi, Gregory Campbell, Wendell D Jones, Fabien Campagne, Zhining Wen, Stephen J Walker, Zhenqiang Su, Tzu-Ming Chu, Federico M Goodsaid, Lajos Pusztai, et al. The microarray quality control (maqc)-ii study of common practices for the development and validation of microarray-based predictive models. *Nature biotechnology*, 28(8): 827, 2010.
70. Maxim V Kuleshov, Matthew R Jones, Andrew D Rouillard, Nicolas F Fernandez, Qiaonan Duan, Zichen Wang, Simon Koplev, Sherry L Jenkins, Kathleen M Jagodnik, Alexander

Lachmann, et al. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic acids research*, 44(W1):W90–W97, 2016.
71. Cristina Aguayo-Mazzucato, Joshua Andle, Terrence B Lee Jr, Ayush Midha, Lindsay Talemal, Vaja Chipashvili, Jennifer Hollister-Lock, Jan van Deursen, Gordon Weir, and Susan Bonner-Weir. Acceleration of β cell aging determines diabetes and senolysis improves disease outcomes. *Cell metabolism*, 2019.
72. Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.
73. Felipe Pezoa, Juan L Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273. International World Wide Web Conferences Steering Committee, 2016.
74. Guido Barbaglia, Simone Murzilli, and Stefano Cudini. Definition of rest web services with json schema. *Software: Practice and Experience*, 47(6):907–920, 2017.
75. Markus Lanthaler and Christian Gütl. On using json-ld to create evolvable restful services. In *Proceedings of the Third International Workshop on RESTful Design*, pages 25–32. ACM, 2012.
76. Sam Newman. *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2015.
77. Meherun Nesa Lucky, Marco Cremaschi, Barbara Lodigiani, Antonio Menolascina, and Flavio De Paoli. Enriching api descriptions by adding api profiles through semantic annotation. In *International Conference on Service-Oriented Computing*, pages 780–794. Springer, 2016.
78. Cristian Gadea, Mircea Trifan, Dan Ionescu, and Bogdan Ionescu. A reference architecture for real-time microservice api consumption. In *Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms*, page 2. ACM, 2016.
79. Hamza Ed-Douibi, Javier Luis Cánovas Izquierdo, and Jordi Cabot. Example-driven web api specification discovery. In *European Conference on Modelling Foundations and Applications*, pages 267–284. Springer, 2017.
80. Oren Ben-Kiki, Clark Evans, and Brian Ingerson. Yaml ain't markup language (yaml™) version 1.1. *yaml. org, Tech. Rep*, page 23, 2005.
81. Amrapali Zaveri, Shima Dastgheib, Chunlei Wu, Trish Whetzel, Ruben Verborgh, Paul Avillach, Gabor Korodi, Raymond Terryn, Kathleen Jagodnik, Pedro Assis, et al. Smartapi: Towards a more intelligent network of web apis. In *European Semantic Web Conference*, pages 154–169. Springer, 2017.
82. Jiten Bhagat, Franck Tanoh, Eric Nzuobontane, Thomas Laurent, Jerzy Orlowski, Marco Roos, Katy Wolstencroft, Sergejs Aleksejevs, Robert Stevens, Steve Pettifer, et al. Biocatalogue: a universal catalogue of web services for the life sciences. *Nucleic acids research*, 38(suppl_2):W689–W694, 2010.
83. Andrew Nightingale, Ricardo Antunes, Emanuele Alpi, Borisas Bursteinas, Leonardo Gonzales, Wudong Liu, Jie Luo, Guoying Qi, Edd Turner, and Maria Martin. The proteins api: accessing key integrated protein and genome information. *Nucleic acids research*, 45(W1): W539–W544, 2017.
84. Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, et al. The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic acids research*, page gkw937, 2016.
85. Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014. ISSN 1075-3583.
86. Di Liu and Libin Zhao. The research and implementation of cloud computing platform based on docker. In *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP)*, pages 475–478. IEEE, 2014.
87. Paolo Di Tommaso, Emilio Palumbo, Maria Chatzou, Pablo Prieto, Michael L Heuer, and Cedric Notredame. The impact of docker containers on the performance of genomic pipelines. *PeerJ*, 3:e1273, 2015.
88. Amos A Folarin, Richard JB Dobson, and Stephen J Newhouse. Ngseasy: a next generation sequencing pipeline in docker containers. *F1000Research*, 4, 2015.
89. Espen Mikal Robertsen, Tim Kahlke, Inge Alexander Raknes, Edvard Pedersen, Erik Kjærner Semb, Martin Ernstsen, Lars Ailo Bongo, and Nils Peder Willassen. Metapipe-pipeline annotation, analysis and visualization of marine metagenomic sequence data. *arXiv preprint arXiv:1604.04103*, 2016.
90. Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, et al. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.
91. Bjørn Fjukstad and Lars Ailo Bongo. A review of scalable bioinformatics pipelines. *Data Science and Engineering*, 2(3):245–251, 2017.
92. Ling-Hong Hung, Daniel Kristiyanto, Sung Bong Lee, and Ka Yee Yeung. Guidock: using docker containers with a common graphics user interface to address the reproducibility of research. *PloS one*, 11(4):e0152686, 2016.
93. Denis Torre, Alexander Lachmann, and Avi Ma'ayan. Biojupies: automated generation of interactive notebooks for rna-seq data analysis in the cloud. *Cell systems*, 7(5):556–561, 2018.
94. Alexander Lachmann, Zhuorui Xie, and Avi Ma'ayan. Elysium: Rna-seq alignment in the cloud. *bioRxiv*, page 382937, 2018.
95. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Supplementary Note 1: Supplementary Material

A. Benchmark of lowly and highly expressed genes. We combine the gene expression matrices from the five aligners into a single matrix. We then apply quantile normalization on the matrix forcing the samples into the same distribution. We then apply a log2 transformation on the quantile normalized gene counts. We first filter out genes that have no to or shallow expression. We remove genes with less than 0.1 log2 counts on average across all datasets. We then compute the quantile ranges in 10% intervals. We use all genes in the 10% to 40% interval as lowly expressed genes and all genes in the 70% to 100% interval as highly expressed. We then compute the Pearson correlation between the gene expression of a gene across samples between pairwise alignment methods.

B. Technical reproducibility of control samples. For each of the 80 signatures, we select the control samples and calculate the pairwise correlation. These correlations should be high compared to correlations of control samples and perturbation samples. To quantify who well control and perturbation samples can be separated for each alignment algorithm, we divide the correlation of the control samples by the average correlation of the perturbation and control sample pairs. If control samples are on average more similar to each other than to perturbation samples, these scores are larger than one.

C. qPCR concordance. In order to measure the concordance of qPCR to RNA-seq derived fold changes we downloaded the qPCR data from the supplementary files of its publication (60). We then processed the 16 RNA-seq samples of the same study with the five alignment algorithms. We then calculated the fold change between MAQCA and MAQCB samples, after log2 transform and quantile normalization. The qPCR data contains outliers with extreme reported fold changes. We identify them by calculating a k-means with 3 clusters. The largest cluster with the smallest absolute fold changes was retained and the genes from the other two clusters was removed from further analysis. The genes from these clusters have no correlation to RNA-seq data and their fold change is most likely high due to technical errors. We calculate the Pearson correlation between the qPCR fold changes and RNA-seq samples for the matching genes. We also compute the average fold change by combining the 5 alignment algorithms as a separate method.

D. Enrichment analysis. We selected gene set libraries GO Biological Process 2018, KEGG 2019 Human, ChEA 2016, and Human Phenotype Ontology from EnrichR containing 5103, 309, 645, and 1779 gene sets for a total of 7836 gene sets. From each differential gene expression signature, we select the top 250 genes with the most significant up-regulation after perturbation. We use the enrichR R package from CRAN to calculate the significance of overlap. For each differential gene expression signature, we select the biological terms that have a corrected p-value < 0.05. For each method, we build a set of all biological terms that were significant in at least one differential gene expression profile and report the cardinality on the diagonal of table 2. We then calculate the size of the intersection of the sets for each pair of the alignment tool.

E. Johnson-Lindenstrauss benchmark. To compute the Johnson-Lindenstrauss transformation nearest-neighbor search we select 1000 random samples from the human ARCHS4 dataset with each sample containing the gene counts of 35238 genes. We then apply quantile normalization followed by a z-score transformation of the genes and calculate the Pearson correlation of the samples. We create a transformation matrix for each dimension from 10 to 5000 dimensions. We then project the original expression matrix to the lower dimensional space and calculate the pairwise correlations of the dimensionally reduced samples. We then calculate the correlation vector of each sample in the full dimensional space and the reduced space.

F. Docker file configuration. Software versions used were STAR 2.7.1a, BWA 0.7.17-r1188, kallisto 0.45.1, Salmon v0.13.1, HISAT2 version 2.1.0 and featureCount Version 1.6.4. The alignment benchmark docker container is preconfigured using conda. The docker image can be loaded with the following command:

```
docker run -it maayanlab/alignmentbenchmark bash
```

The benchmark can be rerun locally or in the cloud with slight modifications to the scripts contained in the scripts folder of the docker container. The alignment script requires at least 20GB of system memory. Scripts are provided to build index files for reference genomes, align multiple SRA as a batch job, load prebuilt indices (recommended as BWA requires around 50 GB of memory), R scripts for building mapping files from transcripts to genes, aggregating transcript counts to gene counts. Some parts of the scripts make use of the AWS CLI. In order to execute these parts of the scripts, the AWS ID and AWS KEY have to be set. please refer to https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html

G. Reference genome. All benchmarks were performed on the Homo sapiens Ensembl GRCh38 with release-96.
ftp://ftp.ensembl.org/pub/release-96/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh38.cdna.all.fa.gz
ftp://ftp.ensembl.org/pub/release-96/gtf/homo_sapiens/Homo_sapiens.GRCh38.96.gtf.gz

H. Alignment parameters. For the aligners we ran the following commands for unpaired samples:

```
STAR \
    --genomeDir "index/star/${SPECIES}_96" \
    --limitBAMsortRAM 10000000000 \
    --runThreadN $thread_num \
    --outSAMstrandField intronMotif \
    --outFilterIntronMotifs RemoveNoncanonical \
    --outFileNamePrefix quant/star/$SRA_FILE/$SRA_FILE \
    --readFilesIn "sradata/${SRA_FILE}.fastq" \
    --outSAMtype BAM SortedByCoordinate \
    --outReadsUnmapped Fastx \
    --outSAMmode Full \
    --quantMode GeneCounts \
    --limitIObufferSize 50000000

salmon quant -i "index/salmon/salmon_${SPECIES}_96" -l A \
    -r "sradata/${SRA_FILE}.fastq" \
    -p $thread_num -q --validateMappings -o quant/salmon/$SRA_FILE

kallisto quant -i index/kallisto/kallisto_${SPECIES}_96.idx \
    -t $thread_num -o quant/kallisto/$SRA_FILE \
    --single -l 180 -s 20 "sradata/${SRA_FILE}.fastq"

hisat2 \
    -x "index/hisat2/${SPECIES}_96/${SPECIES}" \
    -U "sradata/${SRA_FILE}.fastq" \
    -p $thread_num \
    -S "quant/hisat2/${SRA_FILE}/${SRA_FILE}.sam"
featureCounts -T $thread_num -a "reference/${SPECIES}_96.gtf" \
    -o "quant/hisat2/${SRA_FILE}/${SRA_FILE}.tsv" "quant/hisat2/${SRA_FILE}/${SRA_FILE}.sam"

bwa mem \
    -t $thread_num \
    "index/bwa/${SPECIES}_96/${SPECIES}_96" \
    "sradata/${SRA_FILE}.fastq" \
    > "quant/bwa/${SRA_FILE}/${SRA_FILE}.sam"
featureCounts -T $thread_num -a "reference/${SPECIES}_96.gtf" \
    -o "quant/bwa/${SRA_FILE}/${SRA_FILE}.tsv" "quant/bwa/${SRA_FILE}/${SRA_FILE}.sam"
```

For paired-end samples we ran:

```
STAR \
    --genomeDir "index/star/${SPECIES}_96" \
    --limitBAMsortRAM 10000000000 \
    --runThreadN $thread_num \
    --outSAMstrandField intronMotif \
    --outFilterIntronMotifs RemoveNoncanonical \
    --outFileNamePrefix quant/star/$SRA_FILE/$SRA_FILE \
    --readFilesIn "sradata/${SRA_FILE}_1.fastq" "sradata/${SRA_FILE}_2.fastq" \
    --outSAMtype BAM SortedByCoordinate \
    --outReadsUnmapped Fastx \
    --outSAMmode Full \
    --quantMode GeneCounts \
    --limitIObufferSize 50000000

salmon quant -i "index/salmon/salmon_${SPECIES}_96" -l A \
    -1 "sradata/${SRA_FILE}_1.fastq" \
    -2 "sradata/${SRA_FILE}_2.fastq" \
    -p $thread_num -q --validateMappings -o quant/salmon/$SRA_FILE
```

```
kallisto quant -i index/kallisto/kallisto_${SPECIES}_96.idx \
    "sradata/${SRA_FILE}_1.fastq" \
    "sradata/${SRA_FILE}_2.fastq" \
    -t $thread_num -o quant/kallisto/$SRA_FILE

hisat2 \
    -x "index/hisat2/${SPECIES}_96/${SPECIES}" \
    -1 "sradata/${SRA_FILE}_1.fastq" \
    -2 "sradata/${SRA_FILE}_2.fastq" \
    -p $thread_num \
    -S "quant/hisat2/${SRA_FILE}/${SRA_FILE}.sam"
featureCounts -T $thread_num -a "reference/${SPECIES}_96.gtf" \
    -o "quant/hisat2/${SRA_FILE}/${SRA_FILE}.tsv" "quant/hisat2/${SRA_FILE}/${SRA_FILE}.sam"

bwa mem \
    -t $thread_num \
    "index/bwa/${SPECIES}_96/${SPECIES}_96" \
    "sradata/${SRA_FILE}_1.fastq" \
    "sradata/${SRA_FILE}_2.fastq" \
    > "quant/bwa/${SRA_FILE}/${SRA_FILE}.sam"
featureCounts -T $thread_num -a "reference/${SPECIES}_96.gtf" \
    -o "quant/bwa/${SRA_FILE}/${SRA_FILE}.tsv" "quant/bwa/${SRA_FILE}/${SRA_FILE}.sam"
```

I. Data availability. Gene counts:
https://mssm-alignment-benchmark.s3.amazonaws.com/kallisto_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/salmon_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/star_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/hisat2_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/bwa_expression.tsv

Differential Expression Profiles:
https://mssm-alignment-benchmark.s3.amazonaws.com/kallisto_diff_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/salmon_diff_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/star_diff_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/hisat2_diff_expression.tsv
https://mssm-alignment-benchmark.s3.amazonaws.com/bwa_diff_expression.tsv

J. GitHub repositories. https://github.com/MaayanLab/alignmentbenchmark
https://github.com/MaayanLab/biojupies
https://github.com/MaayanLab/elysium
https://github.com/MaayanLab/charon

Supplementary Note 2: Supplementary Figures and Tables

Publication	First (3 points)	Second (2 points)	Third (1 point)	0 points
(55)	Novoalign	CLC	GSNAP	STAR, HISAT
(46)	Bowtie2	kallisto	Sailfish	HISAT, TOPHAT2
(57)	Sailfish	kallisto	STAR	TOPHAT
(51)	HISAT2	STAR	TOPHAT2	GSNAP, OLEGO
(56)	Novoalign	BWA	Bowtie2	
(21)	HISAT2	kallisto, Salmon		
(60)	kallisto, Salmon	HISAT2		
(20)	kallisto, STAR			

Table S1. Benchmark ranking from eight review papers. The ranking was manually created and not always is a clear ranking of methods possible to discern.

Perturbation	Type	Controls	Treatments
Salmonella	Disease	GSM1528198,GSM1528202,GSM1528208	GSM1528199,GSM1528210,GSM1528204
Salmonella	Disease	GSM1528200,GSM1528211,GSM1528205	GSM1528201,GSM1528213,GSM1528207
T0901317	drug	GSM2042924,GSM2042926,GSM2042927	GSM2042931,GSM2042930,GSM2042929
Perhexiline	drug	GSM1612793,GSM1612792,GSM1612791	GSM1612796,GSM1612795,GSM1612794
Vemurafenib	drug	GSM1579148,GSM1579149,GSM1579147	GSM1579152,GSM1579151,GSM1579150
Vemurafenib	drug	GSM1579148,GSM1579149,GSM1579147	GSM1579155,GSM1579154,GSM1579153
Panobinostat	drug	GSM1365937,GSM1365957,GSM1365977	GSM1365978,GSM1365958,GSM1365938
Panobinostat	drug	GSM1365987,GSM1365947,GSM1365967	GSM1365968,GSM1365948,GSM1365988
Panobinostat	drug	GSM1365972,GSM1365932,GSM1365952	GSM1365973,GSM1365933,GSM1365953
Panobinostat	drug	GSM1365942,GSM1365962,GSM1365982	GSM1365943,GSM1365963,GSM1365983
5-Azacytidine	drug	GSM1365937,GSM1365957,GSM1365977	GSM1365979,GSM1365959,GSM1365939
5-Azacytidine	drug	GSM1365987,GSM1365947,GSM1365967	GSM1365969,GSM1365949,GSM1365989
5-Azacytidine	drug	GSM1365972,GSM1365932,GSM1365952	GSM1365954,GSM1365934,GSM1365974
5-Azacytidine	drug	GSM1365942,GSM1365962,GSM1365982	GSM1365984,GSM1365964,GSM1365944
Panobinostat + 5-Azacytidine	drug	GSM1365937,GSM1365957,GSM1365977	GSM1365960,GSM1365940,GSM1365980
Panobinostat + 5-Azacytidine	drug	GSM1365987,GSM1365947,GSM1365967	GSM1365990,GSM1365970,GSM1365950
Panobinostat + 5-Azacytidine	drug	GSM1365972,GSM1365932,GSM1365952	GSM1365955,GSM1365935,GSM1365975
Panobinostat + 5-Azacytidine	drug	GSM1365972,GSM1365932,GSM1365952	GSM1365985,GSM1365965,GSM1365945
NMP	drug	GSM1365937,GSM1365957,GSM1365977	GSM1365961,GSM1365941,GSM1365981
NMP	drug	GSM1365987,GSM1365947,GSM1365967	GSM1365991,GSM1365971,GSM1365951
NMP	drug	GSM1365972,GSM1365932,GSM1365952	GSM1365936,GSM1365956,GSM1365976
NMP	drug	GSM1365942,GSM1365962,GSM1365982	GSM1365986,GSM1365946,GSM1365966
Perhexiline	drug	GSM1612793,GSM1612792,GSM1612791	GSM1612796,GSM1612795,GSM1612794
Erlotinib	drug	GSM1595850,GSM1595851,GSM1595852	GSM1595853,GSM1595854,GSM1595855
Erlotinib	drug	GSM1595858,GSM1595856,GSM1595857	GSM1595859,GSM1595860
Erlotinib	drug	GSM1595864,GSM1595863,GSM1595862	GSM1595865,GSM1595867,GSM1595866
Silvestrol	drug	GSM1370699,GSM1370701,GSM1370700	GSM1370704,GSM1370703,GSM1370702
JAK3i (CP-690550)	drug	GSM1396738,GSM1396736,GSM1396737	GSM1396739,GSM1396741,GSM1396740
SYKi (R406)	drug	GSM1396738,GSM1396736,GSM1396737	GSM1396744,GSM1396743,GSM1396742
JAK3i (CP-690550)	drug	GSM1396721,GSM1396723,GSM1396722	GSM1396725,GSM1396724,GSM1396726
SYKi (R406)	drug	GSM1396721,GSM1396723,GSM1396722	GSM1396729,GSM1396728,GSM1396727
ZNA	drug	GSM1431872,GSM1431871,GSM1431870	GSM1431875,GSM1431874,GSM1431873
ZNA	drug	GSM1431877,GSM1431876,GSM1431878	GSM1431880,GSM1431881,GSM1431879
DAPT 3	drug	GSM1624196,GSM1624197,GSM1624198	GSM1624200,GSM1624201,GSM1624199
LNA 30bcd	drug	GSM2108603,GSM2108604,GSM2108605	GSM2108600,GSM2108601,GSM2108602
LNA 30bcd	drug	GSM2108609,GSM2108612,GSM2108610	GSM2108608,GSM2108606,GSM2108607
LNA 30bcd	drug	GSM2108617,GSM2108616,GSM2108615	GSM2108614,GSM2108613,GSM2108612
Cortistatin A	drug	GSM2071737,GSM2071736,GSM2071738	GSM2071740,GSM2071741,GSM2071739
BZLT	knockdown	GSM2091146,GSM2091147,GSM2091148	GSM2091154,GSM2091153,GSM2091152
GLI1	knockdown	GSM1395357,GSM1395371,GSM1395385	GSM1395391,GSM1395377,GSM1395363
GLI1	knockdown	GSM1395386,GSM1395372,GSM1395358	GSM1395392,GSM1395364,GSM1395378
PSMD13	knockdown	GSM1395386,GSM1395372,GSM1395358	GSM1395381,GSM1395395,GSM1395367
PSMD13	knockdown	GSM1395359,GSM1395373,GSM1395387	GSM1395368,GSM1395382,GSM1395396
PRMT5	knockdown	GSM1945842,GSM1945840,GSM1945841	GSM1945837,GSM1945839,GSM1945838
LPA2	knockdown	GSM1658683,GSM1658682,GSM1658681	GSM1658686,GSM1658685,GSM1658684
NHERF2	knockdown	GSM1658683,GSM1658682,GSM1658681	GSM1658687,GSM1658689,GSM1658688
NAF1	knockdown	GSM1615534,GSM1615535,GSM1615533	GSM1615536,GSM1615537,GSM1615538
PER2	knockdown	GSM1534558,GSM1534559,GSM1534560	GSM1534561,GSM1534563,GSM1534562
OVOL1	knockdown	GSM1635093,GSM1635092,GSM1635091	GSM1635096,GSM1635095,GSM1635094
HER2	knockdown	GSM1466929,GSM1466928,GSM1466930	GSM1466932,GSM1466933,GSM1466931
PRDM11	knockdown	GSM1357458,GSM1357456,GSM1357457	GSM1357453,GSM1357454,GSM1357455
TCF7L2	knockdown	GSM1556984,GSM1556983,GSM1556982	GSM1556987,GSM1556986,GSM1556985
MOV10	overexpression	GSM1220268,GSM1220269,GSM1220267	GSM1220266,GSM1220264,GSM1220265
BRG1	knockdown	GSM1517759,GSM1517758,GSM1517760	GSM1517761,GSM1517762,GSM1517763
MITF	knockdown	GSM1517770,GSM1517768,GSM1517769	GSM1517773,GSM1517772,GSM1517771
MITF	knockdown	GSM1517770,GSM1517768,GSM1517769	GSM1517776,GSM1517775,GSM1517774
TCF21	knockdown	GSM1085737,GSM1085736,GSM1085738	GSM1085740,GSM1085741,GSM1085739
HER2	knockdown	GSM1466929,GSM1466928,GSM1466930	GSM1466932,GSM1466933,GSM1466931
EPAS1	knockdown	GSM1537575,GSM1537574,GSM1537573	GSM1537577,GSM1537576,GSM1537578
JUN	knockdown	GSM1294145,GSM1294146,GSM1294147	GSM1294148,GSM1294149,GSM1294150
JUND	knockdown	GSM1294145,GSM1294146,GSM1294147	GSM1294153,GSM1294152,GSM1294151
CCNK	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369080,GSM1369095,GSM1369065
DDPA5	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369082,GSM1369097
EIF3L	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369068,GSM1369098,GSM1369083
GLI1	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369099,GSM1369069,GSM1369084
JAK2	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369100,GSM1369085
POLR2D	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369102,GSM1369087,GSM1369072
PSMD13	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369099,GSM1369103,GSM1369073
RGS18	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369089,GSM1369074,GSM1369104
SAP130	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369090,GSM1369075,GSM1369105
TGM5	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369076,GSM1369106
TOX4	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369092,GSM1369107
BEGAIN	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369064,GSM1369094,GSM1369079
CDK10	knockdown	GSM1369093,GSM1369063,GSM1369078	GSM1369081,GSM1369096
SENCR	knockdown	GSM1254458,GSM1254459,GSM1254457	GSM1254462,GSM1254461,GSM1254460
SRA	knockdown	GSM1264338,GSM1264339,GSM1264340	GSM1264341,GSM1264343,GSM1264342
PRMT4	knockdown	GSM1122674,GSM1122675,GSM1122673	GSM1122676,GSM1122677,GSM1122678

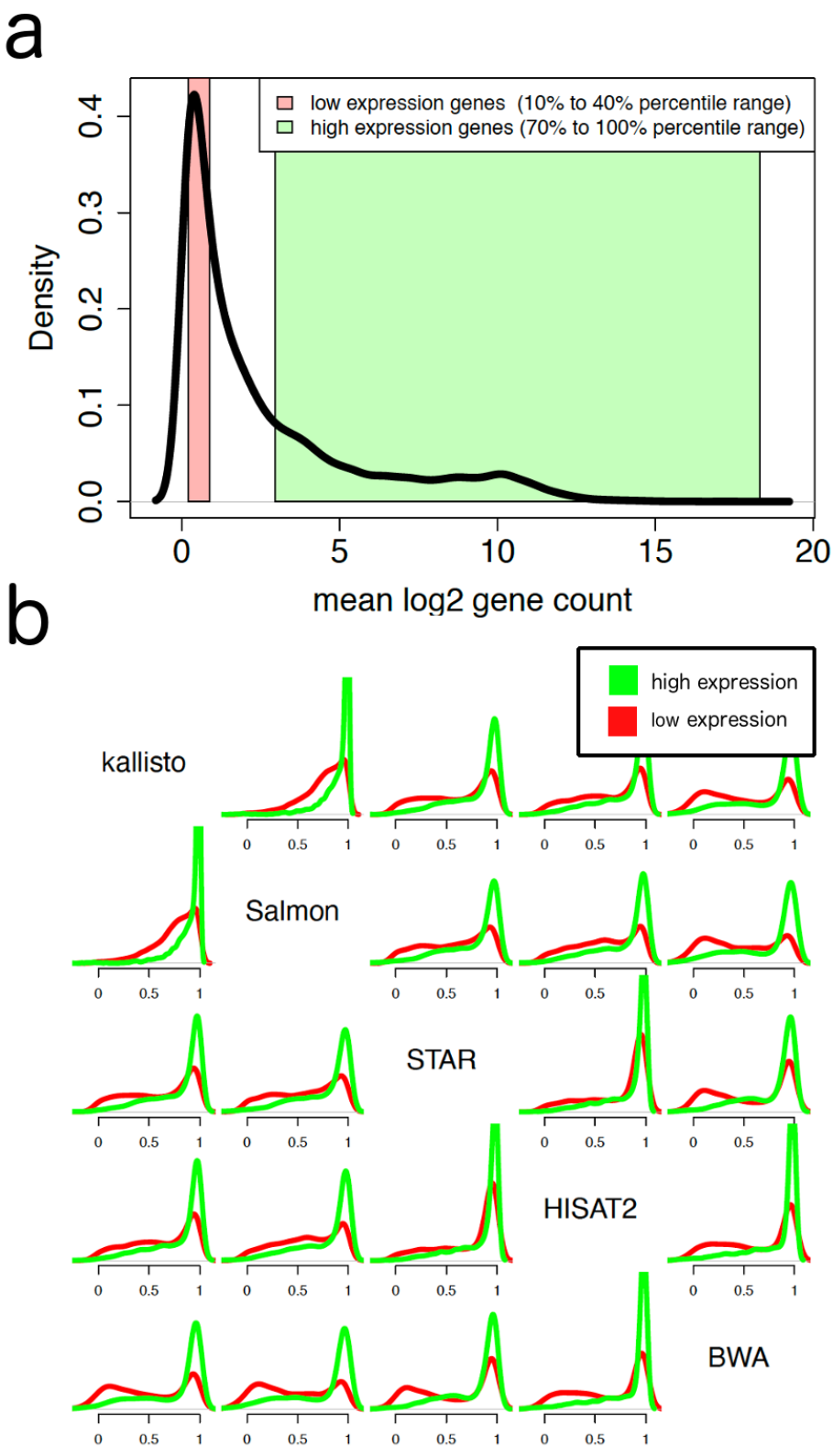

Fig. S1. a) Density plot of average log2 gene counts. The red area highlights the range of genes deemed lowly expressed, and the green represents the highly expressed genes. b) Gene correlation of highly expressed genes in green and lowly expressed genes in red between different alignment methods.

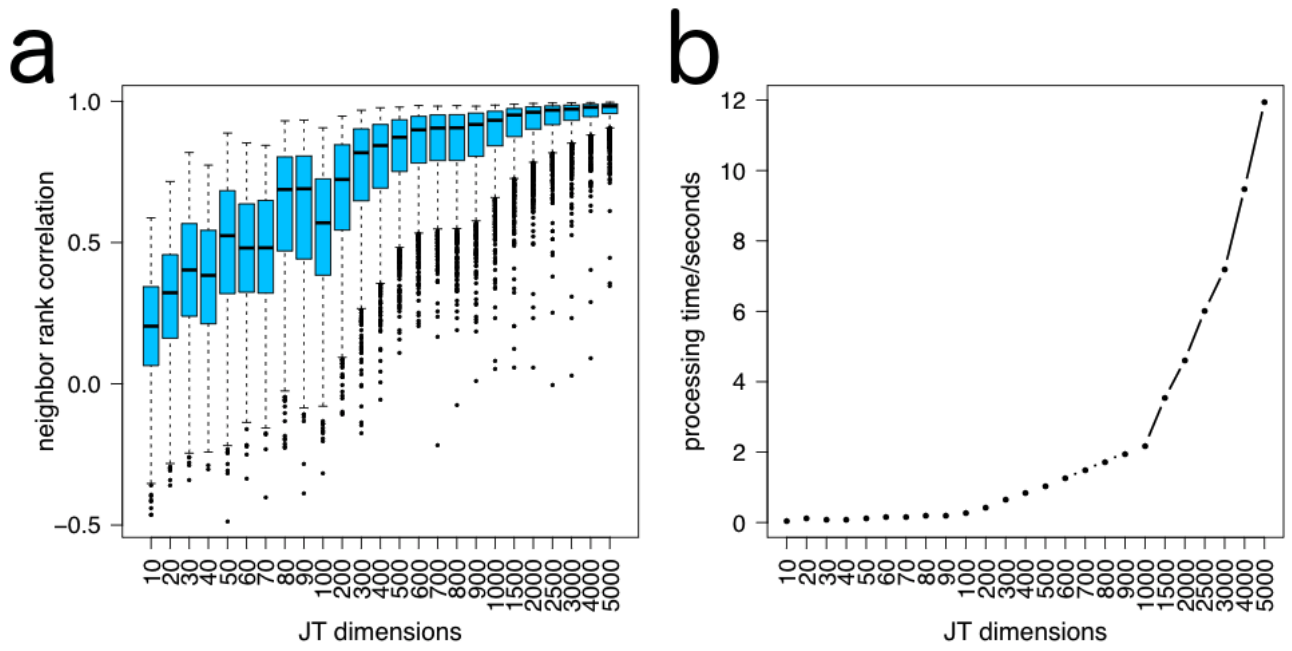

Fig. S2. Johnson-Lindenstrauss Transformation benchmark. a) The boxplot shows the correlation distribution of the original nearest-neighbor ranking produced on full dimensionality search to a nearest-neighbor search in reduced dimensionality space. b) The processing time to perform a nearest-neighbor search on different sized of sub-dimensional space.

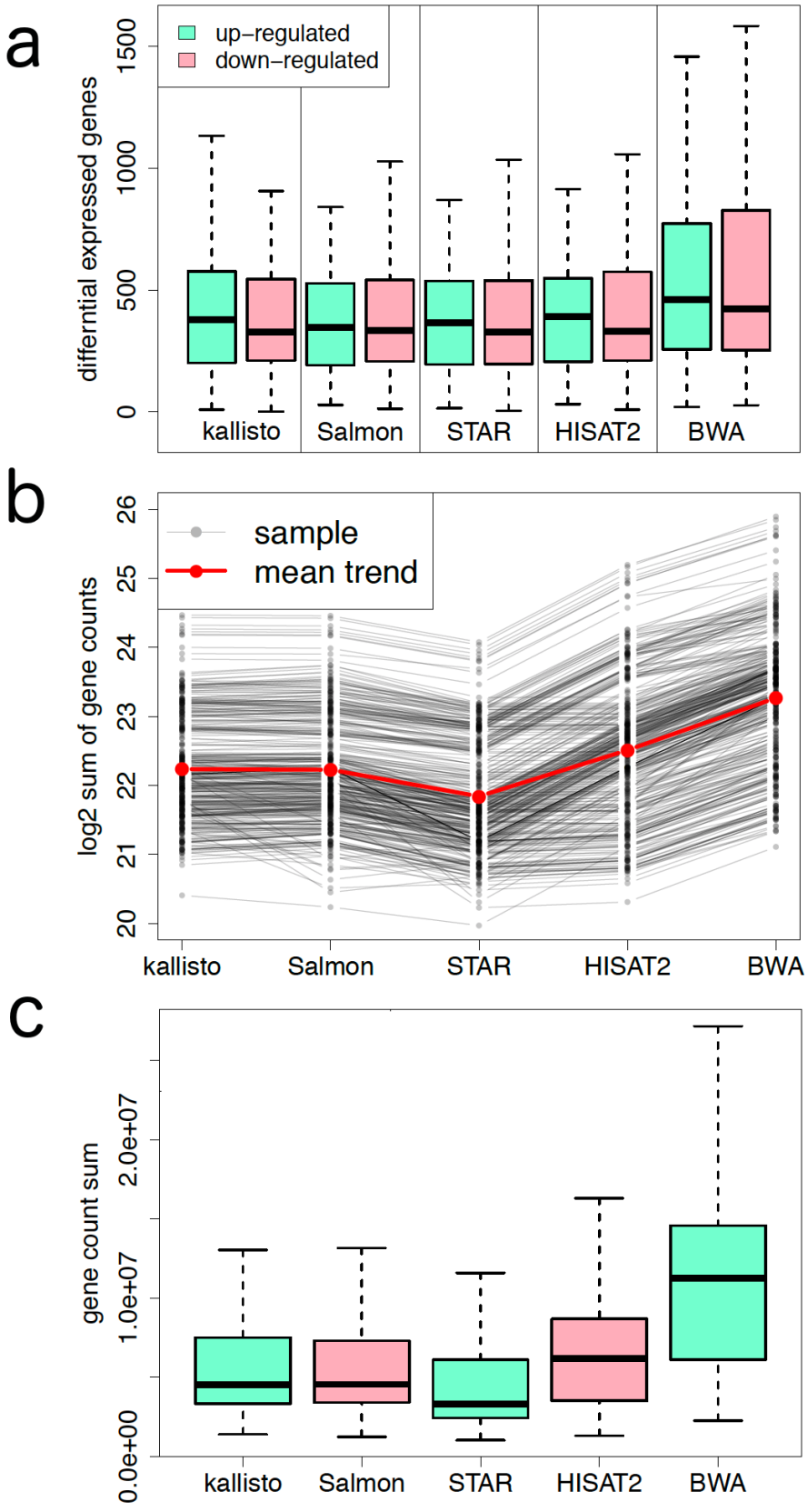

Fig. S3. Alignment methods benchmark. a) Number of up and down regulated genes by alignment algorithm. b) Sum of gene counts per sample and alignment method with each line representing a single sample. c) Distribution of sum of gene counts by alignment algorithm.

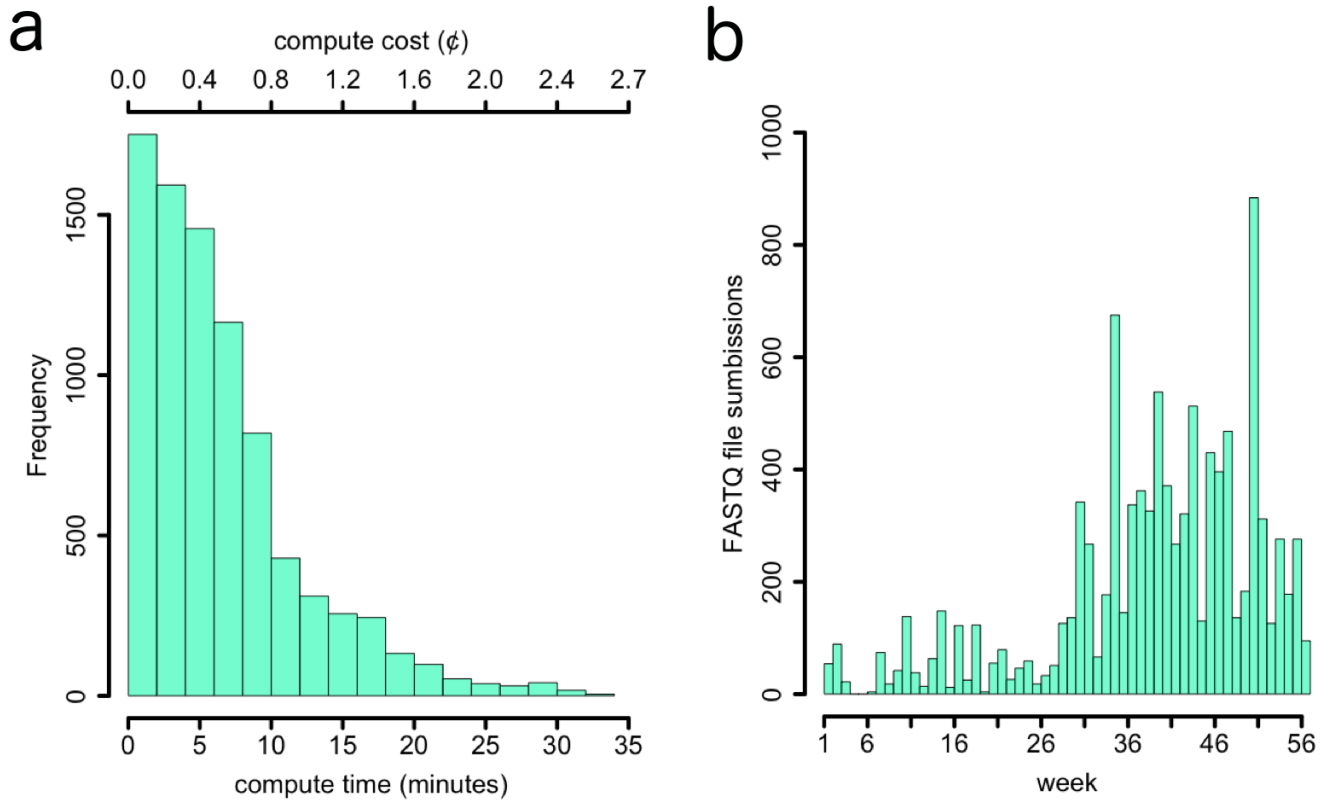

Fig. S4. Elysium cloud alignment pipeline benchmark. a) Histogram of compute time per sample and cost generated by cloud compute hardware. b) Load distribution of Elysium architecture over a span of 56 weeks.